When Distributed Computation does not Help

David P. Woodruff
IBM Almaden
dpwoodru@us.ibm.com

Qin Zhang
IBM Almaden
qinzhang@cs.au.dk

Abstract

We consider a number of basic statistical and graph problems in the message-passing model, where we have k machines (sites), each holding a piece of data, and the machines want to jointly solve a problem defined on the union of the k data sets. The communication is point-to-point, and the goal is to minimize the total communication between the k sites. This model captures all point-to-point distributed computational models in terms of communication costs, including the BSP model and the MapReduce model. Our analysis shows that exact computation of many statistical and graph problems in the distributed setting are very expensive, and often one cannot do better than simply having all machines send their data to a centralized server. Thus, in order to obtain protocols that are communication-efficient, one has to allow approximation, or investigate the distribution or layout of the data sets.

1 Introduction

Recent years have witnessed a spectacular increase in the amount of data being collected and processed in various applications. In many of these applications, the size of the data is too large to fit on a single machine, and thus the data is often distributed across a group of machines, referred to as *sites* in this paper, which are connected by a communication network. These sites jointly compute a function defined on the union of the data sets by exchanging messages with each other. Concrete models for this type of computation include the BSP model [37] and the recent and extensively-studied MapReduce model [12]. Popular system architectures include Hadoop [1], Google's Pregel [29], Microsoft's Trinity [2].

Unlike traditional centralized computation, in which we are mainly concerned with the CPU processing time and the number of disk accesses, in distributed computational models for big data we are more interested in minimizing two objectives, namely, the *communication cost* and the *round complexity*. The communication cost, which we shall also refer to as the communication complexity, denotes the total number of bits exchanged in all messages across the sites during a computation. The round complexity refers to the number of messages, which we shall also refer to as rounds, exchanged in order to complete the computation, without specifying how many bits need to be exchanged in each message.

The communication cost is a fundamental measure since communication is often the bottleneck of applications, and so it directly relates to overall running time, energy consumption, and network bandwidth usage. The round complexity is critical when the computation is partitioned into rounds and the initialization of each round requires a large overhead. In this paper we will focus on the communication complexity, and analyze problems in an abstract model called the message-passing model (see the definition in Section 1.1) that captures all models for point-to-point distributed computation in terms of their communication costs. In particular, our lower bound results hold even if the communication protocol sends only a single bit in each message, and each site has an unbounded amount of local memory and computational power. Note that this

means our lower bounds are as strong as possible, not requiring any assumptions on the local computational power of the machines. We also present several upper bounds, all of which are also locally computationally efficient, meaning the protocols we present do not need extra memory beyond what is required to accommodate the input. We will briefly discuss the issue of round-efficiency in Section 7.

Common sources of massive data include numerical data, e.g., IP streams and logs of queries to a search engine, as well as graph data, e.g., web graphs, social networks, and citation graphs. In this paper we investigate the communication costs for solving several basic statistical and graph problems in the message-passing model. Solving these problems is a minimal requirement of protocols seeking to solve more complicated functions on distributed data.

We show that if we want to solve many of these problems exactly, then there are no better solutions than the almost trivial ones, which are usually quite communication-inefficient. The motivation of this work is thus to deliver the following message to people working on designing protocols for solving problems on distributed databases: for many statistical and graph problems in the distributed setting, if we want efficient communication protocols, then we need to consider the following relaxations to the original problem:

1. Allow for returning an approximate solution. Here, approximation can be defined as follows: for a problem whose output is a single numerical value x, allowing an approximation means that the protocol is allowed to return any value $\tilde{x}$ for which $\tilde{x} \in [(1-\varepsilon)x, (1+\varepsilon)x]$, for some small user-specified parameter $\varepsilon > 0$. For a problem whose output is YES or NO, e.g., a problem deciding if a certain property of the input exists or not, we could instead allow the protocol to return YES if the input is close to having the property (under some problem-specific notion of closeness) and NO if the input is far from having that property. For example, in the graph connectivity problem, we return YES if the graph can be made connected by adding a small number of edges, while we return NO if the graph requires adding a large number of edges to be made connected. This latter notion of approximation coincides with the *property testing* paradigm [18] in the computer science literature.

   By allowing certain approximations we can sometimes drastically reduce the communication costs. Concrete examples and case studies will be given in Section 2 and Section 6.6.

2. Use well-designed input layouts. Here are two examples: (1) All edges from the same node are stored in the same site or on only a few sites. In our lower bounds the edges adjacent to a node are typically stored across many different sites. (2) Each edge is stored on a unique (or a small number) of different sites. Our results in Table 1 show that whether or not the input graph has edges that occur on multiple sites can make a huge difference in the communication costs.

3. Explore prior distributional properties of the input dataset, e.g., if the dataset is skewed, or the underlying graph is sparse or follows a power-law distribution. Instead of developing algorithms targeting the worst-case distributions, as those used in our lower bounds, if one is fortunate enough to have a reasonable model of the underlying distribution of inputs, this can considerably reduce communication costs. An extreme example is that of a graph on n vertices - if the graph is completely random, meaning, each possible edge appears independently with probability p, then the k sites can simply compute the total number of edges m to decide whether or not the input graph is connected with high probability. Indeed, by results in random graph theory, if $m \geq 0.51n \log n$ then the graph is connected with very high probability, while if $m \leq 0.49n \log n$ then the graph is disconnected with very high probability [14]. Of course, completely random graphs are unlikely to appear in practice, though other distributional assumptions may also result in more tractable problems.

1.1 The Message-Passing model

In this paper we consider the message-passing model, studied, for example, in [31, 38]. In this model we have k sites, e.g., machines, sensors, database servers, etc., which we denote as $P_1, \ldots, P_k$. Each site has some portion of the overall data set, and the sites would like to compute a function defined on the union of the k data sets by exchanging messages. The communication is point-to-point, that is, if P_i talks to P_j, then the other $k-2$ sites do not see the messages exchanged between P_i and P_j. At the end of the computation, at least one of the sites should report the correct answer. The goal is to minimize the total number of bits and messages exchanged among the k sites. For the purposes of proving impossibility results, i.e., lower bounds, we can allow each site to have an infinite local memory and infinite computational power; note that such an assumption will only make our lower bounds stronger. Further, we do not place any constraints on the format of messages or any ordering requirement on the communication, as long as it is point-to-point.

The message-passing model captures all point-to-point distributed communication models in terms of the communication cost, including the BSP model by Valiant [37], the $\mathcal{MRC}$ MapReduce framework proposed by Karloff et al. [25], the generic MapReduce model by Goodrich et al. [19], and the Massively Parallel model by Koutris and Suciu [26].

1.2 Our Results

We investigate lower bounds (impossibility results) and upper bounds (protocols) of the exact computation of the following basic statistical and graph problems in the message-passing model.

1. Statistical problems: computing the number of distinct elements, known as F_0 in the database literature; and finding the element with the maximum frequency, known as the ℓ_∞ or iceberg query problem. We note that the lower bound for ℓ_∞ also applies to the heavy-hitter problem of finding all elements whose frequencies exceed a certain threshold, as well as many other statistical problems for which we have to compute the elements with the maximum frequency exactly.

2. Graph problems: computing the degree of a vertex; testing cycle-freeness; testing connectivity; computing the number of connected components (#CC); testing bipartiteness; and testing triangles-freeness.

For each graph problem, we study its lower bound and upper bound in two cases: with edge duplication among the different sites and without edge duplication. Our results are summarized in Table 1. Note that all lower bounds are matched by upper bounds up to some logarithmic factors. For convenience, we use $\tilde{\Omega}(f)$ and $\tilde{O}(f)$ to denote functions of forms $f/\log^{O(1)}(f)$ and $f \cdot \log^{O(1)}(f)$, respectively. That is, we hide logarithmic factors.

We prove most of our lower bound results via reductions from a meta-problem that we call THRESH_θ^r . Its definition is given in Section 4.

In Section 6.6 we make a conjecture on the lower bound for the diameter problem, i.e., the problem of computing the distance of the farthest pair of vertices in a graph. This problem is one of the few problems that we cannot completely characterize by the technique proposed in this paper. We further show that by allowing an error as small as an additive-2, we can reduce the communication cost of computing the diameter by roughly a $\sqrt{n}$ factor, compared with the naive algorithm for exact computation. This further supports our claim that even a very slight approximation can result in a dramatic savings in communication.

	With duplication		Without duplication	
Problem	LB	UB	LB	UB
F_0	$\tilde{\Omega}(kF_0)$	$\tilde{O}(k(F_0 + \log n))$	–	–
ℓ_∞	$\tilde{\Omega}(\min\{k, \ell_\infty\}n)$	$\tilde{O}(\min\{k, \ell_\infty\}n)$	–	–
degree	$\tilde{\Omega}(kd_v)$	$O(kd_v \log n)$	$\tilde{\Omega}(k)$	$O(k \log n)$
cycle-freeness	$\tilde{\Omega}(kn)$	$\tilde{O}(kn)$	$\Omega(\min\{n, m\})$	$\tilde{O}(\min\{n, m\})$
connectivity	$\tilde{\Omega}(kn)$	$\tilde{O}(kn)$	$\tilde{\Omega}(kr)$	$\tilde{O}(kr)$
#CC	$\tilde{\Omega}(kn)$	$\tilde{O}(kn)$	$\tilde{\Omega}(kr)$	$\tilde{O}(kr)$
bipartiteness	$\tilde{\Omega}(kn)$	$\tilde{O}(kn)$	$\tilde{\Omega}(kr)$	$\tilde{O}(kr)$
triangle-freeness	$\tilde{\Omega}(km)$	$\tilde{O}(km)$	$\Omega(m)$	$\tilde{O}(m)$

Table 1: All results are in terms of number of bits of communication. Our lower bounds hold for randomized protocols which succeed with at least a constant probability of $2/3$, while all of our upper bounds are deterministic protocols (which always succeed). k refers to the number of sites, with a typical value ranging from 100 to 10000 in practice. For F_0 and ℓ_∞, n denotes the size of the element universe. For graph problems, n denotes the number of vertices and m denotes the number of edges. d_v is the degree of the queried vertex v. We make the mild assumption that $\Omega(\log n) \leq k \leq \min\{n, m\}$. Let $r = \min\{n, m/k\}$. Except for the upper bound for cycle-freeness in the "without duplication" case, for which $m \geq n$ implies that a cycle necessarily exists (and therefore makes the problem statement vacuous), we assume that $m \geq n$ in order to avoid a messy and uninteresting case-by-case analysis.

1.3 Related Work

Quite a few graph problems have been recently studied in the MapReduce model, including computing the connected components [28, 32], counting the number of triangles [35, 4], finding a minimum spanning tree [28], computing a matching and an edge cover [28], finding a minimum-cut [28], and finding densest subgraphs [7]. The primary goal of all these works is to minimize the round complexity, given various constraints on the number of messages that each site can send/receive at each round. The algorithm in [32] for finding connected components requires $O(\log n)$ rounds and $\tilde{\Omega}((n + m))$ bits of communication per round. For triangle-counting, the total amount of computation (including the local running time of each site) in [35] is $\tilde{O}(m^{3/2})$. For matching and edge cover, certain forms of approximation are needed in [28], from which it is shown that the total computational time is $\tilde{O}(m)$. In this line of research, an implicit assumption is generally made that there is no edge duplication in the input across the different sites.

Ahn, Guha and McGregor [5, 6] developed an elegant technique for *sketching* graphs, and showed its applicability to many graph problems including connectivity, bipartiteness, and minimum spanning tree. Each sketching step in these algorithms can be implemented in the message-passing model as follows: each site computes a sketch of its local graph and sends its sketch to P_1. The site P_1 then combines these k sketches into a sketch of the global graph. The final answer can be obtained based on the global sketch that P_1 computes. Most sketches in [5, 6] are of size $\tilde{O}(n^{1+\gamma})$ bits (for a small constant $\gamma \geq 0$), and the number of sketching steps varies from 1 to a constant. Thus direct implementations of these algorithms in the message-passing model have communication $\tilde{O}(k \cdot n^{1+\gamma})$ bits. A good property of these sketching algorithms is that the sketches can be constructed and maintained in the streaming setting – the edges of the local input graphs can arrive on the site one at a time, without requiring the site to maintain its entire local input in memory. Therefore, these sketching algorithms are quite useful for processing massive graphs.

For statistical problems, a number of approximation algorithms have been proposed recently in the

distributed streaming model, which can be thought of as a dynamic version of the one-shot distributed computation model considered in this paper: the k local inputs arrive in the streaming fashion and one of the sites has to continuously monitor a function defined on the union of the k local inputs. All protocols in the distributed streaming model are also valid protocols in our one-shot computational model, while our impossibility results in our one-shot computational model also apply to all protocols in the distributed streaming model. Example functions studied in the distributed streaming model include F_0 [11], F_2 (size of self join) [11, 38], quantile and heavy-hitters [21]. All of these problems have much lower communication cost if one allows an approximation of the output number x in a range $[(1-\varepsilon)x, (1+\varepsilon)x]$, as mentioned above (the definition as to what ε is for the various problems differs). These works show that if an approximation is allowed, then all these problems can be solved using only $\tilde{O}(k/\varepsilon^{O(1)})$ bits of communication.

On the lower bound side, it seems not much is known for graph problems. Phillips et al. [31] proved an $\Omega(kn/\log^2 k)$ bits lower bound for connectivity. Their lower bound proof relies on a well-crafted graph distribution. In this paper we improve their lower bound by a factor of $\log k$. Another difference is that their proof requires the input to have edge duplications, while our lower bound holds even if there are no edge duplications, showing that the problem is hard even if each edge occurs on a single site. Very recently in an unpublished manuscript, Huang et. al. [20] showed that $\Omega(kn)$ bits of communication is necessary in order to even compute a constant factor approximation to the size of the maximum matching of a graph. Their result, however, requires that the entire matching has to be reported, and it is unknown if a similar lower bound applies if one is only interested in estimating the matching size.

For statistical problems, a suite of lower bounds for approximate computations was developed in [38]. The problems considered in that work include the number F_0 of distinct elements, the p-th frequency moment and self-join size, the heavy-hitters problem, the quantiles and the empirical entropy. For exact F_0 computation, the best previous communication cost lower bound was $\Omega(F_0 + k)$ bits. In this paper we improve the communication cost lower bound to $\tilde{\Omega}(kF_0)$, which is optimal up to a small logarithmic factor. We remark that computing F_0 exactly also allows one to determine whether an item is missing from the union of the k data sets, a problem whose complexity has been studied in the data streaming literature, see, e.g., [36].

Besides statistical and graph problems, Koutris and Suciu [26] studied evaluating conjunctive queries in their massively parallel model. Their lower bounds are restricted to one round of communication, and the message format has to be tuple-based. We stress that in our message-passing model there is no such restriction on the number of rounds and the message format; our lower bounds apply to *arbitrary* communication protocols. Recently, Daumé III et al. [22, 23] and Balcan et al. [8] studied several problems in the setting of distributed learning, in the message-passing model.

1.4 Conventions

Let $[n] = \{1, \ldots, n\}$. All logarithms are base-2. All communication complexities are in terms of bits. We typically use capital letters $X, Y, \ldots$ for sets or random variables, and lower case letters $x, y, \ldots$ for specific values of the random variables $X, Y, \ldots$. We write $X \sim \mu$ to denote a random variable chosen from distribution μ. For convenience we often identify a set $X \subseteq [n]$ with its characteristic vector when there is no confusion, i.e., the bit vector which is 1 in the i-th bit if and only if element i occurs in the set X.

Let $H(X)$ denote the usual Shannon entropy of a random variable X, that is, $H(X) = \sum_x \mathsf{Pr}[X = x] \log(1/\mathsf{Pr}[X = x])$. And let $H(X \mid Y)$ denote the conditional entropy, that is, $H(X \mid Y) = \sum_y \mathsf{Pr}[Y = y] H(X \mid Y = y)$. Let $H_b(p)$ denote the binary entropy function when $p \in [0, 1]$, that is, $H_b(p) = p \log(1/p) + (1-p) \log(1/(1-p))$ with the usual convention that $H_b(0) = H_b(1) = 0$.

1.5 Roadmap

In Section 2, we give a case study on the number of distinct elements (F_0) problem. In Section 3, we include background on communication complexity which is needed for understanding the rest of the paper. In Section 4, we introduce the meta-problem THRESH_θ^r and study its communication complexity. In Section 5 and Section 6, we show how to prove lower bounds for a set of statistical and graph problems by performing reductions from THRESH_θ^r. We conclude the paper in Section 7.

2 The Number of Distinct Elements: A Case Study

In this section we give a case study on the number of distinct elements (F_0) problem, with the purpose of justifying the statement that approximation is often needed in order to obtain communication-efficient protocols in the distributed setting.

The F_0 problem requires computing the number of distinct elements of a data set. For example, given a table *supplier* containing attributes *supplier-id*, *supplier-name* and *city*, if we want to count the number of different cities in the table, we can write the following SQL statement:

SELECT COUNT(DISTINCT city) as "Distinct Cities"
FROM suppliers;

This problem has numerous applications in query optimization [34], data mining in graph databases [30], network traffic monitoring [15], data integration [10], data warehousing [3], and many other database and networking areas.

The F_0 problem has been extensively studied in both the database and theory communities in the last three decades, mainly in the data stream model. It began with the work of Flajolet and Martin [17] and culminated in an optimal algorithm by Kane et al. [24]. In the streaming setting, we see a stream of elements coming one at a time and the goal is to compute the number of distinct elements in the stream using as little memory as possible. In [16], Flajolet et al. reported that their HyperLogLog algorithm can estimate cardinalities beyond 10^9 using a memory of only 1.5KB, and achieve a relative accuracy of 2%, compared with the 10^9 bytes of memory required if we want to compute F_0 exactly.

Similar situations happen in the distributed communication setting, where we have k sites, each holding a set of elements from the universe $[n]$, and the sites want to compute the number of distinct elements of the union of their k data sets. In [11], a $(1+\varepsilon)$-approximation algorithm (protocol) with $O(k(\log n + 1/\varepsilon^2 \log 1/\varepsilon))$ bits of communication was given in the distributed streaming model, which is also a protocol in the message-passing model. In a typical setting, we could have $\varepsilon = 0.01$, $n = 10^9$ and $k = 1000$, in which case the communication cost is about 6.6×10^7 bits [1]. On the other hand, our result shows that if exact computation is required, then the communication cost among the k sites needs to be at least be $\Omega(kF_0/\log k)$ (See Corollary 1), which is already 10^9 bits even when $F_0 = n/100$.

3 Preliminaries

In this section we introduce some background on communication complexity. We refer the reader to the book by Kushilevitz and Nisan [27] for a more complete treatment.

In the basic two-party communication complexity model, we have two parties (also called sites or players), which we denote by Alice and Bob. Alice has an input x and Bob has an input y, and they want to

[1] In the comparison we neglect the constants hidden in the big-O and big-Ω notation which should be small.

jointly compute a function $f(x, y)$ by communicating with each other according to a protocol Π. Let $\Pi(x, y)$ be the transcript of the protocol, that is, the concatenation of the sequence of messages exchanged by Alice and Bob, given the inputs x and y. In this paper when there is no confusion, we abuse notation by using Π for both a protocol and its transcript, and we further abbreviate the transcript $\Pi(x, y)$ by Π.

The *deterministic communication complexity* of a deterministic protocol is defined to be $\max\{|\Pi(x, y)| \mid \text{all possible inputs } (x, y)\}$, where $|\Pi(x, y)|$ is the number of bits in the transcript of the protocol Π on inputs x and y. The *randomized communication complexity* of a randomized protocol Π is the maximum number of bits in the transcript of the protocol over all possible inputs x, y, together with all possible random tapes of the players. We say a randomized protocol Π computes a function f correctly with error probability δ if for all input pairs (x, y), it holds that $\mathsf{Pr}[\Pi(x, y) \neq f(x, y)] \leq \delta$, where the probability is taken only over the random tapes of the players. The randomized *δ-error communication complexity* of a function f, denoted by $R^{\delta}(f)$, is the minimum communication complexity of a protocol that computes f with error probability at most δ.

Let μ be a distribution over the input domain, and let $(X, Y) \sim \mu$. For a deterministic protocol Π, we say that Π computes f with error probability δ on μ if $\mathsf{Pr}[\Pi(X, Y) \neq f(X, Y)] \leq \delta$, where the probability is over the choices of $(X, Y) \sim \mu$. The *δ-error μ-distributional communication complexity* of f, denoted by $D^{\delta}_{\mu}(f)$, is the minimum communication complexity of a deterministic protocol that computes f with error probability δ on μ. We denote $\mathsf{E}[D^{\delta}_{\mu}(f)]$ to be the expected distributional communication complexity where the expectation is taken over the input distribution μ.

We can generalize the two-party communication complexity to the multi-party setting, which is the message-passing model considered in this paper. Here we have k players (also called sites) $P_1, \ldots, P_k$ with P_j having the input x_j, and the players want to compute a function $f(x_1, \ldots, x_k)$ of their joint inputs by exchanging messages with each other. The transcript of a protocol always specifies which player speaks next. In this paper the communication is point-to-point, that is, if P_i talks to P_j, the other players do not see the messages sent from P_i to P_j. At the end of the communication, only one player needs to output the answer.

The following lemma shows that randomized communication complexity is lower bounded by distributional communication complexity under any distribution μ.

Lemma 1 (one direction of Yao's Lemma [39]) *For any function f and any $\delta > 0$,*

$$R^{\delta}(f) \geq \max_{\mu} D^{\delta}_{\mu}(f).$$

Proof: The original proof is for two players, though this also holds for $k > 2$ players since for any distribution μ, if Π is a δ-error protocol then for all possible inputs $x^1, \ldots, x^k$ to the k players,

$$\mathsf{Pr}_{\text{random tapes of the players}}[\Pi(x^1, \ldots, x^k) = f(x^1, \ldots, x^k)] \geq 1 - \delta,$$

which implies for any distribution μ on $(x^1, \ldots, x^k)$ that

$$\mathsf{Pr}_{\text{random tapes of the players}, (x^1, \ldots, x^k) \sim \mu}[\Pi(x^1, \ldots, x^k) = f(x^1, \ldots, x^k)] \geq 1 - \delta,$$

which implies there is a fixing of the random tapes of the players so that

$$\mathsf{Pr}_{(x^1, \ldots, x^k) \sim \mu}[\Pi(x^1, \ldots, x^k) = f(x^1, \ldots, x^k)] \geq 1 - \delta,$$

which implies $D^{\delta}_{\mu}(f)$ is at most $R^{\delta}(f)$. ■

Therefore, one way to prove a lower bound on the randomized communication complexity of f is to first pick a (hard) input distribution μ for f, and then study its distributional communication complexity under μ.

Note that given a $1/3$-error randomized protocol for a problem f whose output is 0 or 1, we can always run the protocol $C\log(1/\delta)$ times using independent randomness each time, and then output the majority of the outcomes. By a standard Chernoff bound (see below), the output will be correct with error probability at most $e^{-\kappa C\log(1/\delta)}$ for an absolute constant κ, which is at most δ if we choose C to be a sufficiently large constant. Therefore $R^{1/3}(f) = \Omega(R^{\delta}(f)/\log(1/\delta)) = \Omega(\max_{\mu} D^{\delta}_{\mu}(f)/\log(1/\delta))$ for any $\delta \in (0, 1/3]$. Consequently, to prove a lower bound on $R^{1/3}(f)$ we only need to prove a lower bound on the distributional communication complexity of f with an error probability $\delta \leq 1/3$.

Chernoff bound. Let $X_1, \ldots, X_n$ be independent Bernoulli random variables such that $\mathsf{Pr}[X_i = 1] = p_i$. Let $X = \sum_{i\in[n]} X_i$. Let $\mu = \mathsf{E}[X]$. It holds that $\mathsf{Pr}[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3}$ and $\mathsf{Pr}[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2}$ for any $\delta \in (0, 1)$.

4 A Meta-Problem

In this section we discuss a meta-problem THRESH^r_θ and we derive a communication lower bound for it. This meta-problem will be used to derive lower bounds for statistical and graph problems in our applications.

In the THRESH^r_θ problem, site P_i ($i \in [k]$) holds an r-bit vector $x_i = \{x_{i,1}, \ldots, x_{i,r}\}$, and the k sites want to compute

$$\text{THRESH}^r_\theta(x_1, \ldots, x_k) = \begin{cases} 0, & \text{if } \sum_{j\in[r]}(\vee_{i\in[k]} x_{i,j}) \leq \theta, \\ 1, & \text{if } \sum_{j\in[r]}(\vee_{i\in[k]} x_{i,j}) \geq \theta + 1. \end{cases}$$

That is, if we think of the input as a $k \times r$ matrix with $x_1, \ldots, x_k$ as the rows, then in the THRESH^r_θ problem we want to find out whether the number of columns that contain a 1 is more than θ for a threshold parameter θ.

We will show a lower bound for THRESH^r_θ using the symmetrization technique introduced in [31]. First, it will be convenient for us to study the problem in the *coordinator* model.

The Coordinator Model. In this model we have an additional site called the coordinator [2], which has no input (formally, his input is the empty set). We require that the k sites can only talk to the coordinator. The message-passing model can be simulated by the coordinator model since every time a site P_i wants to talk to P_j, it can first send the message to the coordinator, and then the coordinator can forward the message to P_j. Such a re-routing only increases the communication complexity by a factor of 2 and thus will not affect the asymptotic communication complexity.

Let $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be an arbitrary function. Let μ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$. Let $f^k_{\text{OR}} : \mathcal{X}^k \times \mathcal{Y} \to \{0, 1\}$ be the problem of computing $f(x_1, y) \vee f(x_2, y) \vee \ldots \vee f(x_k, y)$ in the coordinator model, where P_i has input $x_i \in \mathcal{X}$ for each $i \in [k]$, and the coordinator has $y \in \mathcal{Y}$. Given the distribution μ on $\mathcal{X} \times \mathcal{Y}$, we construct a corresponding distribution ν on $\mathcal{X}^k \times \mathcal{Y}$: We first pick $(X_1, Y) \sim \mu$, and then pick $X_2, \ldots, X_k$ from the conditional distribution $\mu \mid Y$.

[2] We can also choose, for example, P_1 to be the coordinator and avoid the need for an additional site, though having an additional site makes the notation cleaner.

The following theorem was originally proposed in [31]. Here we improve it by a $\log k$ factor by a slightly modified analysis, which we include here for completeness.

Theorem 1 *For any function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ and any distribution μ on $\mathcal{X} \times \mathcal{Y}$ for which $\mu(f^{-1}(1)) \leq 1/k^2$, we have $D_{\nu}^{1/k^2}(f_{\mathrm{OR}}^k) = \Omega(k \cdot \mathsf{E}[D_{\mu}^{1/(100k)}(f)])$.*

Proof: Suppose Alice has X and Bob has Y with $(X,Y) \sim \mu$, and they want to compute $f(X,Y)$. They can use a protocol $\mathcal{P}$ for f_{OR}^k to compute $f(X,Y)$ as follows. The first step is an input reduction. Alice and Bob first pick a random $I \in [k]$ using shared randomness, which will later be fixed by the protocol to make it deterministic. Alice simulates P_I by assigning it an input $X_I = X$. Bob simulates the coordinator and the remaining $k-1$ players. He first assigns Y to the coordinator, and then samples $X_1, \ldots, X_{I-1}, X_{I+1}, \ldots, X_k$ independently according to the conditional distribution $\mu \mid Y$, and assigns X_i to P_i for each $i \in [k] \backslash I$. Now $\{X_1, \ldots, X_k, Y\} \sim \nu$. Since $\mu(f^{-1}(1)) \leq 1/k^2$, with probability $(1-1/k^2)^{k-1} \geq 1 - 1/k$, we have $f(X_i, Y) = 0$ for all $i \in [k] \backslash I$. Consequently,

$$f_{\mathrm{OR}}^k(X_1, \ldots, X_k, Y) = f(X,Y). \tag{1}$$

We say such an input reduction is *good*.

Alice and Bob construct a protocol $\mathcal{P}'$ for f by independently repeating the input reduction twice, and running $\mathcal{P}$ on each input reduction. The probability that at least one of the two input reductions is good is at least $1 - 1/k^2$, and Bob can learn which reduction is good without any communication. This is because in the simulation he locally generates all X_i ($i \in [k] \backslash I$) together with Y. On the other hand, by a union bound, the probability that $\mathcal{P}$ is correct for both two input reductions is at least $1 - 2/k^2$. Note that if we can compute $f_{\mathrm{OR}}^k(X_1, \ldots, X_k, Y)$ correctly for a good input reduction, then by (1), $\mathcal{P}$ can also be used to correctly compute $f(X,Y)$. Therefore $\mathcal{P}$ can be used to compute $f(X,Y)$ with probability at least $1 - 2/k^2 - 1/k^2 \geq 1 - 1/(100k)$.

Since in each input reduction, $X_1, \ldots, X_k$ are independent and identically distributed, and since $I \in [k]$ is chosen randomly in the two input reductions, we have that in expectation over the choice of I, the communication between P_I and the coordinator is at most a $2/k$ fraction of the expected total communication of $\mathcal{P}$ given inputs drawn from ν. By linearity of expectation, if the expected communication cost of $\mathcal{P}$ for solving f_{OR}^k under input distribution ν with error probability at most $1/k^2$ is C, then the expected communication cost of $\mathcal{P}'$ for solving f under input distribution μ with error probability at most $1/(100k)$ is $O(C/k)$. Finally, by averaging there exists a fixed choice of $I \in [k]$, so that $\mathcal{P}'$ is deterministic and for which the expected communication cost of $\mathcal{P}'$ for solving f under input distribution μ with error probability at most $1/(100k)$ is $O(C/k)$. Therefore we have $D_{\nu}^{1/k^2}(f_{\mathrm{OR}}^k) = \Omega(k \cdot \mathsf{E}[D_{\mu}^{1/(100k)}(f)])$. ■

4.1 The 2-DISJ^r Problem

Now we choose a concrete function f to be the set-disjointness problem. In this problem we have two parties: Alice has $x \subseteq [r]$ while Bob has $y \subseteq [r]$, and the parties want to compute

$$\text{2-DISJ}^r(x,y) = \begin{cases} 1, & \text{if } x \cap y \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

Set-disjointness is a classical problem used in proving communication lower bounds. For example, in a database context it was recently used in proving streaming lower bounds for finding dense subgraphs [7]. We define an input distribution τ_β for 2-DISJ^r as follows. Let $\ell = (r+1)/4$. With probability β, x and y

are random subsets of $[r]$ such that $|x| = |y| = \ell$ and $|x \cap y| = 1$, while with probability $1 - \beta$, x and y are random subsets of $[r]$ such that $|x| = |y| = \ell$ and $x \cap y = \emptyset$. Razborov [33] proved that for $\beta = 1/4$, $D^{(1/4)/100}_{\tau_{1/4}}(\text{2-DISJ}^r) = \Omega(r)$, and one can extend his arguments to any $\beta \in (0, 1/4]$, and to the expected distributional communication complexity where the expectation is take over the input distribution.

Theorem 2 ([31], Lemma 2.2) *For any $\beta \in (0, 1/4]$, it holds that $\mathsf{E}[D^{\beta/100}_{\tau_\beta}(\text{2-DISJ}^r)] = \Omega(r)$, where the expectation is taken over the input distribution.*

4.2 The OR-DISJ^r Problem

If we choose f to be 2-DISJ^r and let $\mu = \tau_\beta$, then we call f^k_{OR} in the coordinator model the OR-DISJ^r Problem. By Theorem 1 and Theorem 2. We have

Theorem 3 $D^{1/k^2}_\nu(\text{OR-DISJ}^r) = \Omega(kr)$.

4.2.1 The Complexity of THRESH$^r_\theta$

We prove our lower bound for the setting of the parameter $\theta = (3r - 1)/4$. We define the following input distribution ζ for THRESH$^r_{(3r-1)/4}$: We choose $\{X_1, \ldots, X_k, Y\} \sim \nu$ where ν is the input distribution for OR-DISJ^r, and then simply use $\{X_1, \ldots, X_k\}$ as the input for THRESH$^r_\theta$.

Lemma 2 *Under the distribution ζ, assuming $k \geq c_k \log r$ for a large enough constant c_k, we have that $\bigvee_{i \in [k]} X_{i,j} = 1$ for all $j \in [r] \backslash Y$ with probability $1 - 1/k^{10}$.*

Proof: For each $j \in [r] \backslash Y$, we have $\bigvee_{i \in [k]} X_{i,j} = 1$ with probability at least $1 - (1 - 1/4)^k$. This is because $\mathsf{Pr}[X_{i,j} = 1] \geq 1/4$ for each $j \in [r] \backslash Y$, by our choices of X_i. By a union bound, with probability at least

$$\begin{aligned} & \Big(1 - (3/4)^k \cdot |[r] \backslash Y|\Big) \\ = \; & \Big(1 - (3/4)^k \cdot (3r - 1)/4\Big) \\ \geq \; & 1 - 1/k^{10} \end{aligned}$$

(by our assumption $c_k \log r \leq k \leq r$ for a large enough constant c_k), we have $\bigvee_{i \in [k]} X_{i,j} = 1$ for all $j \in [r] \backslash Y$. ∎

Theorem 4 $D^{1/k^3}_\zeta(\text{THRESH}^r_{(3r-1)/4}) = \Omega(kr)$, *assuming $c_k \log r \leq k \leq r$ for a large enough constant c_k.*

Proof: By Lemma 2, it is easy to see that any protocol $\mathcal{P}$ that computes THRESH$^r_{(3r-1)/4}$ on input distribution ζ correctly with error probability $1/k^3$ can be used to compute OR-DISJ^r on distribution ν correctly with error probability $1/k^3 + 1/k^{10} < 1/k^2$, since if $(X_1, \ldots, X_k, Y) \sim \nu$, then with probability $1 - 1/k^{10}$, we have

$$\text{OR-DISJ}^r(X_1, \ldots, X_k, Y) = (\exists j \in Y, \text{2-DISJ}^r(X_j, Y) = 1) = \text{THRESH}^r_{(3r-1)/4}(X_1, \ldots, X_k).$$

The theorem follows from Theorem 3. ∎

5 Statistical Problems

For technical convenience in the reductions, we make the mild assumption that $c_k \log n \leq k \leq n$ where c_k is some large enough constant. For convenience, we will repeatedly ignore an additive $O(1/k^{10})$ error probability introduced in the reductions, since these will not affect the correctness of the reductions, and can be added to the overall error probability by a union bound.

5.1 F_0 (#distinct-elements)

Recall that in the F_0 problem, each site P_i has a set $S_i \subseteq [n]$, and the k sites want to compute the number of distinct elements in $\bigcup_{i\in[k]} S_i$.

For the lower bound, we reduce from $\text{THRESH}^n_{(3n-1)/4}$. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^n_{(3n-1)/4}$, each site sets $S_i = X_i$. Let σ_F be the input distribution of F_0 after this reduction.

By Lemma 2 we know that under distribution ζ, with probability $1 - 1/k^{10}$, for all $j \in [n]\backslash Y$ (recall that Y is the random subset of $[n]$ of size $(n+1)/4$ we used to construct $X_1, \ldots, X_k$ in distribution ζ), $\bigvee_{i\in[k]} X_{i,j} = 1$. Conditioned on this event, we have

$$\begin{aligned} & \text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 1 \\ \iff \quad & F_0(\cup_{i\in[k]} S_i) > (3n-1)/4. \end{aligned}$$

Therefore, by Theorem 4 we have that $D^{1/k^3}_{\sigma_F}(F_0) = \Omega(kn)$. Note that in this reduction, we have to choose $n = \Theta(F_0)$. Therefore, it makes more sense to write the lower bound as $D^{1/k^3}_{\sigma_F}(F_0) = \Omega(kF_0)$.

The following corollary follows from Yao's Lemma (Lemma 1) and the discussion following it.

Corollary 1 $R^{1/3}(F_0) = \Omega(kF_0/\log k)$*, assuming* $c_k \log F_0 \leq k \leq F_0$ *for a large enough constant* c_k*.*

Similar corollaries hold for all other problems in Section 5 and Section 6, using Yao's Lemma, and we will not explicitly state such corollaries.

An almost matching upper bound of $O(k(F_0 \log F_0 + \log n))$ can be obtained as follows: the k sites first compute a 2-approximation F_0' to F_0 using the protocol in [11] (see Section 2), which costs $O(k \log n)$ bits. Next, they hash every element to a universe of size $(F_0')^3$, so that there are no collisions among hashed elements with probability at least $1 - 1/F_0$, by a union bound. Finally, all sites send their distinct elements (after hashing) to P_1 and then P_1 computes the number of distinct elements over the union of the k sets locally. This step costs $O(kF_0 \log F_0)$ bits of communication.

5.2 ℓ_∞ (MAX)

In the ℓ_∞ problem, each site P_i has a set $S_i \subseteq [n]$, and the k sites want to find an element in $\bigcup_{i\in[k]} S_i$ with the maximum frequency.

For the lower bound, we again reduce from $\text{THRESH}^n_{(3n-1)/4}$. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^n_{(3n-1)/4}$, the k sites create an input $\{S_1, \ldots, S_k\}$ as follows: first, P_1 chooses a set $R \subseteq [k]$ by independently including each $i \in [k]$ with probability $7/8$, and informs all sites P_i ($i \in R$) by sending each of them a bit. This step costs $O(k)$ bits of communication. Next, for each $i \in R$, P_i flips $X_{i,j}$ for each $j \in [n]$. Finally, each P_i includes $j \in S_i$ if $X_{i,j} = 1$ after the flip and $j \notin S_i$ if $X_{i,j} = 0$. Let σ_L be the input distribution of ℓ_∞ after this reduction.

They repeat this input reduction independently T times where $T = c_T \log k$ for a large constant c_T, and at each time they run $\ell_\infty(\cup_{i\in[k]} S_i)$. Let $R_1, \ldots, R_T$ be the random set R sampled by P_1 in the T runs, and

let $O_1, \ldots, O_T$ be the outputs of the T runs. They return $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 1$ if there exists a $t \in [T]$ such that $O_t \geq |R_t| + 1$ and 0 otherwise.

We focus on a particular input reduction. We view an input for $\text{THRESH}^n_{(3n-1)/4}$ as a $k \times n$ matrix. The i-th row of the matrix is X_i. After the bit-flip operations, for each column $j \in [n]\backslash Y$, we have for each $i \in [k]$ that

$$\begin{aligned} & \mathsf{Pr}[X_{i,j} = 1] \\ \leq \; & 7/8 \cdot \left(1 - \frac{(n+1)/4 - 1}{(3n-1)/4}\right) + 1/8 \cdot \frac{(n+1)/4}{(3n-1)/4} \\ < \; & 3/4. \end{aligned}$$

By a Chernoff bound, for each $j \in [n]\backslash Y$, $\sum_{i \in [k]} X_{i,j} < 13k/16$ with probability $1 - e^{-\Omega(k)}$. Therefore with probability at least $(1 - e^{-\Omega(k)} \cdot n) \geq (1 - 1/k^{10})$ (assuming that $c_k \log n \leq k \leq n$ for a large enough constant c_k), $\sum_{i \in [k]} X_{i,j} < 13k/16$ holds for all $j \in [n]\backslash Y$.

Now we consider columns in Y. We can show again by Chernoff bound that $|R| > 13k/16$ with probability $(1 - 1/k^{10})$ for all columns in Y, since each $i \in [k]$ is included into R with probability $7/8$, and before the flips, the probability that $X_{i,j} = 1$ for an i when $j \in Y$ is negligible. Therefore with probability $(1 - 1/k^{10})$, the column with the maximum number of 1s is in the set Y, which we condition on in the rest of the analysis.

In the case when $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 1$, then with probability at least $1/8$, there exists a column $j \in Y$ and a row $i \in [k]\backslash R$ for which $X_{i,j} = 1$. If this happens, then for this j we have $\sum_{i \in [k]} X_{i,j} \geq |R|+1$, or equivalently, $\ell_\infty(\cup_{i \in [k]} S_i) \geq |R|+1$. Otherwise, if $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 0$, then $\sum_{i \in [k]} X_{i,j} = |R|$ for all $j \in Y$. Therefore, if $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 1$, then the probability that there exists a $t \in [T]$ such that $O_t \geq |R_t| + 1$ is at least $1 - (1 - 1/8)^T > 1 - 1/k^{10}$ (by choosing c_T large enough). Otherwise, if $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 0$, then $O_t = |R_t|$ for all $t \in [T]$.

Since our reduction only uses $T \cdot O(k) = O(k \log k)$ extra bits of communication and introduces an extra error of $O(1/k^{10})$, which will not affect the correctness of the reduction. By Theorem 4, we have that $D^{1/k^3}_{\sigma_L}(\ell_\infty) = \Omega(kn)$. Note that in the reduction, we have to assume that $\Theta(\ell_\infty) = \Theta(k)$. In other words, if $\ell_\infty \ll k$ then we have to choose $k' = \Theta(\ell_\infty)$ sites out of the k sites to perform the reduction. Therefore it makes sense to write the lower bound as $D^{1/k^3}_{\sigma_L}(\ell_\infty) = \Omega(\min\{\ell_\infty, k\}n)$.

Finally, a simple protocol in which all sites send their elements-counts to the first site solves ℓ_∞ with $O(\min\{k, \ell_\infty\}n \log n)$ bits of communication, which is almost optimal in light of our lower bound above.

6 Graph Problems

In this section we consider graph problems. Let $G = (V, E)$ with $|V| = n$ and $|E| = m$ be an undirected graph. Each site has a subgraph $G_i \subseteq G$, and the k sites want to compute a property of G via a communication protocol. For technical convenience we again assume that $c_k \log n \leq k \leq \min\{n, m\}$, where c_k is a large enough constant. Except for the upper bound for cycle-freeness in the without edge duplication case, for which $m \geq n$ always causes the graph to not be cycle-free, we assume that $m \geq n$ to avoid an uninteresting case-by-case analysis.

Most lower bounds in this section are shown by reductions from $\text{THRESH}^r_{(3r-1)/4}$ for some value $r \leq n^2$. For convenience of presentation, during some reductions we may generate graphs with more than n

vertices. This will not affect the order of the lower bounds as long as the number of vertices is $O(n)$ and the number of edges is $O(m)$ (if m appears in the lower bound as a parameter).

The following procedure will be used several times in our reductions. Thus, we present it separately.

Reconstructing Y from $X_1, X_2, \ldots, X_k$. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^r_{(3r-1)/4}$ to the k sites, the first site P_1 can construct Y correctly with probability $1 - O(1/k^{10})$, using $O(r \log r)$ bits of communication, assuming that $c_k \log r \leq k \leq r$ for a large enough constant c_k. We view the input as a $k \times r$ matrix with the k sites' inputs as rows. For each column $j \in [r]$, P_1 randomly selects $c_Y \log r$ sites for some large enough constant c_Y, asks each of them for the j-th bit of their input vectors, and then computes the sum of these bits, denoted by s_j. Note that if $j \in [r] \backslash Y$, then by a Chernoff bound with probability $1 - e^{-\kappa \cdot c_Y \cdot \log r}$ (κ is an absolute constant), we have that $s_j \geq c_Y \log r/2$. Therefore with probability $1 - e^{-\kappa \cdot c_Y \cdot \log r} \cdot r \geq 1 - 1/k^{10}$, for all $j \in [r] \backslash Y$, it holds that $s_j \geq c_Y \log r/2$. On the other hand, again by a Chernoff bound we have that with probability $1 - 1/k^{10}$, for all $j \in Y$, $s_j < c_Y \log r/2$. Therefore P_1 can reconstruct Y correctly with probability $1 - O(1/k^{10})$. Since the $O(r \log r)$ extra bits of communication and the $O(1/k^{10})$ additional error probability will not affect the correctness of any of our reductions below, we can simply assume that P_1 can always reconstruct Y for free.

6.1 Degree

In the degree problem, give a vertex $v \in V$, the k sites want to compute the degree of v.

If edge duplication is not allowed, then the degree problem can be solved in $O(k \log n)$ bits of communication: each site sends the number of edges containing the query vertex to the first site P_1 and then P_1 adds up these k numbers. A lower bound of $\Omega(k)$ bits also holds since each site has to speak at least once in our communication model.

When we allow edge duplication, the degree problem is essentially the same as that of F_0, by the following reduction. Given an input $\{X_1, \ldots, X_k\} \sim \tau_F$ for F_0, the k sites construct a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ of size n. Each site P_i ($i \in [k]$) does the following: for each element $j \in X_i$ and $j \neq 1$, it creates an edge (v_1, v_j). Let G be the resulting graph. Then

$$F_0(X_1, \ldots, X_k) = \text{Degree}(v_1) - 1.$$

Thus, we obtain a lower bound of $\Omega(kd_v)$ bits for the degree problem, where d_v is the degree of the queried vertex v. An $O(kd_v \log n)$ bit upper bound is the following: each site sends all the neighbouring vertices of v to the first site.

6.2 Cycle-freeness

In the cycle-freeness problem, the k sites want to check whether G contains a cycle.

6.2.1 Without Edge Duplication

If edge duplication is not allowed, then we have the following simple protocol: $P_2, \ldots, P_k$ send the number of their local edges to P_1 and P_1 computes the total number of edges in the graph G, denoted by m. If $m \geq n$ then P_1 determines immediately that G contains a cycle, since every graph on n vertices having at least n edges must contain a cycle. Otherwise if $m < n$, then $P_2, \ldots, P_k$ send all their edges to P_1, who then does a local check. The communication cost of this protocol never exceeds $O(k \log n + \min\{m, n\} \log n) = O(\min\{m, n\} \log n)$ bits.

Let $h = \min\{m, n\}$. An $\Omega(h)$ bit lower bound holds even when $k = 2$, by a reduction from the 2-DISJ^h problem: suppose P_1 has X and P_2 has Y, where $(X, Y) \sim \tau_{1/4}$ is the hard input distribution for 2-DISJ^h. P_1 and P_2 construct a graph G on the vertex set $\{s, t, v_1, \ldots, v_h\}$ as follows: for each $i \in X$, P_1 creates an edge (s, v_i), and he/she also creates an additional edge (s, t). Similarly, for each $i \in Y$, P_2 creates an edge (v_i, t). Let σ_{C_1} be the resulting input distribution of G. It is easy to see from the reduction that if $X \cap Y \neq \emptyset$, then there is a cycle in the form of $s \to t \to v_i \to s$ for some $i \in [h]$. Otherwise the graph is a forest. Therefore,

$$\text{2-DISJ}^h(X, Y) = 1 \iff G \text{ contains a cycle.}$$

Therefore by Theorem 2 it follows that $D^{1/400}_{\sigma_{C_1}}(\text{cycle-freeness}) = \Omega(h)$.

6.2.2 With Edge Duplication

For the lower bound, we reduce from $\text{THRESH}^n_{(3n-1)/4}$. For each $j \in [n]$, G contains a vertex v_j. G also contains a special vertex u. The total number of vertices in G is $n + 1$.

Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^n_{(3n-1)/4}$, the k sites create a graph G for cycle-freeness as follows. Each P_i creates an edge (u, v_j) for each $X_{i,j} = 1$. In addition, P_1 reconstructs Y, picks an arbitrary set of $\bar{Y} \subset [n]\backslash Y$ of size $(n + 1)/4$, and creates an arbitrary perfect matching between Y and $\bar{Y}$. Let σ_{C_2} be the resulting input distribution of G. By Lemma 2, we have with probability $(1 - 1/k^{10})$ that all pairs (u, v_j) for $j \in [n]\backslash Y$ are connected. It is easy to see from the reduction that if $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 1$, then there is a cycle of the form $s \to v_i \to v_j \to s$ for some $i \in [n]\backslash Y$ and $j \in Y$. Otherwise the graph is a forest. Therefore,

$$\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 0 \iff G \text{ is cycle-free.}$$

Thus by Theorem 4, we have that $D^{1/k^3}_{\sigma_{C_2}}(\text{cycle-freeness}) = \Omega(kn)$.

There is again a trivial $O(kn \log n)$ upper bound. Each site first checks its local graph and reports directly if a cycle is found, otherwise the site sends all its edges (there are no more than $n - 1$ such edges for a cycle-free graph) to P_1. Finally P_1 checks the cycle-freeness on the union of those edges.

6.3 Connectivity and #CC

In the connectivity problem, the k sites want to check whether G is connected. In the #connected-components (#CC) problem, the k sites want to compute the number of connected components in G. Note that solving #CC also solves connectivity, thus we only show the lower bound for connectivity.

6.3.1 Without Edge Duplication

For the lower bound, we reduce from $\text{THRESH}^r_{(3r-1)/4}$ where $r = \min\{m/k, n\}$. For each $i \in [k]$, G contains a vertex u_i; and for each $j \in [r]$, G contains a vertex v_j. The total number of vertices in G is $n + k \leq 2n$. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^r_{(3r-1)/4}$, the k sites create a graph G for connectivity as follows. Each P_i creates an edge (u_i, v_j) for each $X_{i,j} = 1$. In addition, P_1 reconstructs Y, and then creates a path containing $\{v_j \mid j \in Y\}$ and a path containing $\{v_j \mid j \in [r]\backslash Y\}$. See Figure 1 for an illustration. Let σ_{N_1} be the resulting input distribution of G. It is easy to see from the reduction that

$$\text{THRESH}^r_{(3r-1)/4}(X_1, \ldots, X_k) = 1 \iff G \text{ is connected.}$$

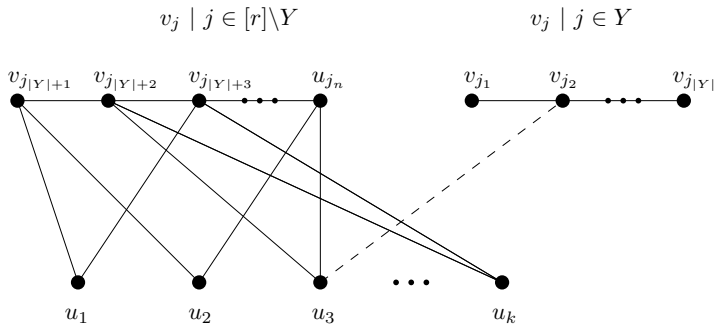

Figure 1: Graph G in the reduction for connectivity.

Thus by Theorem 4, we have that $D^{1/k^3}_{\sigma_{N_1}}(\text{connectivity}) = \Omega(kr)$.

For the upper bound, all connected components (thus #CC and connectivity) can be found by the protocol in which all sites send their local spanning trees to the first site P_1 and then P_1 does a local computation, which costs $O(kr \log n)$ bits of communication.

6.3.2 With Edge Duplication

For the lower bound, we use a slightly modified reduction of the one for the without edge duplication case. We reduce from $\text{THRESH}^n_{(3n-1)/4}$. For each $j \in [n]$, G contains a vertex v_j. G also contains a special vertex u. The total number of vertices in G is $n + 1$. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^n_{(3n-1)/4}$, the k sites create a graph G for connectivity as follows. Each P_i creates an edge (u, v_j) for each $X_{i,j} = 1$. In addition, P_1 reconstructs Y, and then creates a path containing $\{v_j \mid j \in Y\}$ and a path containing $\{v_j \mid j \in [n] \backslash Y\}$. The total number of edges is $O(n) = O(m)$. This graph can be seen as the graph we constructed for the without edge duplication case after merging $u_1, \ldots, u_k$ to a single vertex u while maintaining all the adjacent edges. Let σ_{N_2} be the resulting input distribution of G. The correctness of the reduction is the same as before, that is, $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 1$ if and only if G is connected. Thus by Theorem 4, we have that $D^{1/k^3}_{\sigma_{N_2}}(\text{connectivity}) = \Omega(kn)$.

The upper bound is the same as the without edge duplication case, and the cost is $O(kn \log n)$ bits (note that since we allow edge duplication here, the total number of edges of the k spanning trees cannot be bounded by $O(m)$).

6.4 Bipartiteness

In the bipartiteness problem, the k sites want to check whether G is bipartite.

6.4.1 Without Edge Duplication

For the lower Bound, we reduce from $\text{THRESH}^r_{(3r-1)/4}$ where $r = \min\{m/k, n\}$. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^r_{(3r-1)/4}$, the k sites create a graph $G = (V, E)$ with $V = A \cup B \cup C$ where $A = \{a_1, \ldots, a_r\}$, $B = \{b_1, \ldots, b_r\}$ and $C = \{c_1, \ldots, c_k\}$, as follows: each P_i creates an edge (c_i, b_j) for each $X_{i,j} = 1$. In addition, P_1 does the following:

1. Creates an edge (a_i, b_i) for each $i \in [r]$.

2. Reconstructs Y. For each $i \in [k]$ and $j \in Y$, creates an edge (c_i, a_j).

The total number of vertices of G is $2r + k < 3n$. Let σ_{B_1} be the resulting input distribution of G. One can see from the reduction that if $\text{THRESH}^r_{(3r-1)/4}(X_1, \ldots, X_k) = 1$, then there exists at least one triangle in the form of (a_i, b_i, c_j) for some $i \in [r], j \in [k]$. Otherwise if

$$\text{THRESH}^r_{(3r-1)/4}(X_1, \ldots, X_k) = 0,$$

then all edges are between two vertex sets $\{a_i \mid i \in [r] \backslash Y\} \cup C \cup \{b_i \mid i \in Y\}$ and $\{b_i \mid i \in [r] \backslash Y\} \cup \{a_i \mid i \in Y\}$ and consequently G is a bipartite graph. Therefore,

$$\text{THRESH}^r_{(3r-1)/4}(X_1, \ldots, X_k) = 0 \iff G \text{ is bipartite.}$$

Thus by Theorem 4, we have that $D^{1/k^3}_{\sigma_{B_1}}(\text{bipartiteness}) = \Omega(kr)$.

For the upper bound, we can assume that the graph is connected, since otherwise we can first compute all connected components (which costs $O(kn \log n)$ bits of communication as mentioned in Section 6.3), and then work on each connected component. The protocol works as follows: the first site P_1 chooses an arbitrary vertex u in the graph, and grows a breadth-first-search (BFS) tree rooted at u by communicating with the other $k - 1$ sites. In the first round, P_1 asks each site to report the vertices adjacent to u using its local edges. The communication is at most $O(|N(u)| \log n \cdot k)$ bits, where $N(u)$ denotes the set of neighbors of u, and $|N(u)|$ denotes the number of (distinct) neighbors of u. From this, P_1 computes the entire set $N(u)$ of neighbors of u, without duplication, and sends it to the other $k - 1$ sites. This also takes $O(|N(u)| \log n \cdot k)$ bits of communication. Now the sites all know $N(u)$, and they can build the first layer of the BFS tree rooted at u. Next, P_1 picks the first child v (according to an arbitrary but fixed order) of u, and repeats this process on v. If ever a site finds an odd cycle, it is announced to P_1. Notice that every vertex is sent at most k times to P_1, meanwhile the total number of vertices sent is no more than $O(m)$, so the total communication is $O(kr \log n)$ bits.

6.4.2 With Edge Duplication

We reduce from $\text{THRESH}^n_{(3n-1)/4}$. The reduction is a simple modification of the one for the without edge duplication case. Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^n_{(3n-1)/4}$, the k sites create a graph $G = (V, E)$ with $V = A \cup B \cup C$ where $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ and $C = \{c\}$, as follows: each P_i creates an edge (c, b_j) for each $X_{i,j} = 1$. In addition, P_1 creates an edge (a_i, b_i) for each $i \in [n]$, reconstructs Y, and for each $j \in Y$ creates an edge (c, a_j). The total number of vertices of G is $2n+1$. Let σ_{B_2} be the resulting input distribution of G. The correctness of the reduction is similar as before, that is, $\text{THRESH}^n_{(3n-1)/4}(X_1, \ldots, X_k) = 0$ if and only if G is bipartite. Thus by Theorem 4, we have that $D^{1/k^3}_{\sigma_{B_2}}(\text{bipartiteness}) = \Omega(kn)$.

The upper bound is the same as the without edge duplication case, and the communication complexity is $O(kn \log n)$ bits. Note that since we have edge duplication here, the claim that "the total number of vertices sent is no more than $O(m)$" does not hold.

6.5 Triangle-freeness

In the triangle-freeness problem, the k sites want to check whether G contains a triangle.

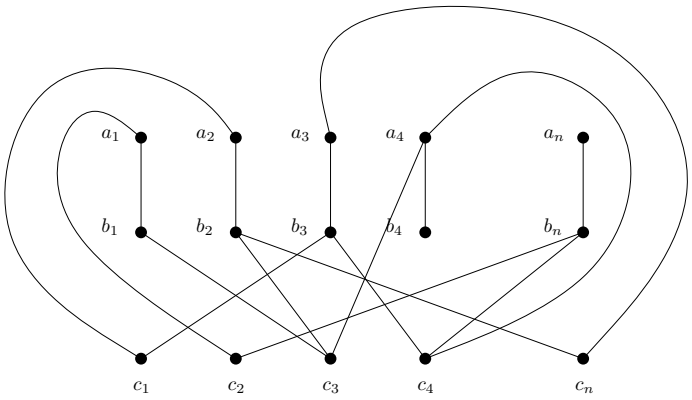

Figure 2: Graph G in the reduction for triangle-freeness.

6.5.1 Without Edge Duplication

An $O(m \log n)$ upper bound is the following: $P_2, \ldots, P_k$ send all their edges to P_1 and then P_1 does a local check.

There is an $\Omega(m)$ bit lower bound on the communication which holds even when $k = 2$, by a reduction from 2-DISJ^m. Suppose P_1 holds X and P_2 holds Y, where $\{X, Y\} \sim \tau_{1/4}$ is the hard input distribution for 2-DISJ^m. Sites P_1 and P_2 construct a graph $G = (V, E)$ with $V = A \cup B \cup C$ where $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ and $C = \{c_1, \ldots, c_n\}$ as follows: for each $i \in X$, P_1 creates an edge (a_p, c_q) such that $(p-1)n + q = i$ $(p, q \in [n])$ (note that the solution of (p, q) is unique). He/she also creates an edge (a_t, b_t) for all $t \in [n]$. Similarly, for each $i \in Y$, P_2 creates an edge (b_p, c_q) such that $(p-1)n + q = i$ $(p, q \in [n])$. The graph G has $3n$ vertices and $O(m)$ edges. See Figure 2 for an illustration. Let σ_{T_1} be the resulting input distribution of G. One can see from the reduction that if $X \cap Y \neq \emptyset$, then there is a triangle in the form of (a_p, b_p, c_q) for some $p, q \in [n]$. Otherwise the graph is triangle-free. Therefore,

$$\text{2-DISJ}^m(X, Y) = 0 \iff G \text{ is triangle free.}$$

Therefore by Theorem 2 we have $D^{1/400}_{\sigma_{T_1}}(\text{triangle-freeness}) = \Omega(m)$.

6.5.2 With Edge Duplication

We reduce from $\text{THRESH}^m_{(3m-1)/4}$, and prove an $\Omega(km)$ lower bound on the communication cost. The reduction is an extension of the one for the without edge duplication case.

Given an input $\{X_1, \ldots, X_k\} \sim \zeta$ for $\text{THRESH}^m_{(3m-1)/4}$, the k sites create the following input graph $G = (V, E)$ for triangle-freeness with $V = A \cup B \cup C$ where $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ and $C = \{c_1, \ldots, c_n\}$. Each site P_i does the following: for each $j \in [m]$ such that $X_{i,j} = 1$, the site creates an edge (a_p, c_q) such that $(p-1)n + q = j$ $(p, q \in [n])$. In addition, the first site P_1 also does the following.

1. Creates an edge (a_t, b_t) for each $t \in [n]$.
2. Reconstructs Y. For each $j \in Y$, creates an edge (b_p, c_q) such that $(p-1)n + q = j$ $(p, q \in [n])$.

Let σ_{T_2} be the resulting input distribution of G. As before, it is easy to see that if $\text{THRESH}^m_{(3m-1)/4}(X_1, \ldots, X_k) = 1$, then there is a triangle of the form (a_p, b_p, c_q) for some $p, q \in [n]$. Otherwise the graph is triangle-free. Therefore,

$$\text{THRESH}^m_{(3m-1)/4}(X_1, \ldots, X_k) = 0 \iff G \text{ is triangle-free.}$$

By Theorem 4, we have that $D^{1/k^3}_{\sigma_{T_2}}(\text{triangle-freeness}) = \Omega(km)$.

There is a simple protocol with $O(km \log n)$ bits of communication: each site sends all its edges to P_1 and the P_1 does a local check.

We comment that our upper and lower bounds also applies to testing clique-freeness, that is, the k sites want to check whether G contains a clique of size s for a fixed constant s.

6.6 A Conjecture on the Diameter Problem and an Approximation Algorithm

We would like to mention the diameter problem which cannot be solved by the technique introduced in this paper. In the diameter problem, the k sites want to compute the diameter of a graph $G = (V, E)$ in which the edges are distributed amongst the k sites. We conjecture the following:

Conjecture 1 *The randomized communication complexity of the diameter problem in the message-passing model is $\tilde{\Omega}(km)$ bits, assuming edge duplication is allowed.*

Note that the naive algorithm in which every site sends all of its edges to the first site will match this lower bound up to a logarithmic factor.

In [13] an algorithm for constructing a graph spanner in the RAM model with an additive distortion 2 is proposed. A graph spanner with an additive distortion d preserves all pairwise distances of vertices in the original graph up to an additive error d. This algorithm can be easily implemented in the message-passing model with a communication complexity of $O(kn^{3/2} \log^2 n)$ bits. This fact gives us a hint that in order to prove an $\tilde{\Omega}(kn^2)$ (in the case when $m = \Theta(n^2)$) lower bound, we have to explore the difficulty of distinguishing a diameter T verses $T+1$ for a value $T \in [1, n-2]$ (in fact, one can show that T also needs to be a fixed constant). Now we briefly describe how to implement the algorithm in [13] in the message-passing model.

The algorithm in [13] works as follows (in the RAM model).

1. We pick $\Theta(\sqrt{n} \log n)$ vertices in the graph uniformly at random, and grow a breadth-first-search (BFS) tree rooted on each of these vertices. We then include all edges of these BFS trees into the spanner.

2. We include all edges incident to vertices whose degrees are no more than $\sqrt{n}$.

In [13] it was shown that this algorithm computes a spanner with an additive distortion 2 correctly with probability 0.99. Now let us briefly discuss how to implement this algorithm in the message-passing model. For the first step, the random sampling can be done by P_1 locally, and then P_1 communicates with the other $k-1$ sites to grow a BFS tree rooted on each of these vertices using the algorithm described in Section 6.4. Recall that constructing each of these BFS trees costs $O(kn \log n)$ bits of communication. Thus the total communication needed for the first step is $O(kn^{3/2} \log^2 n)$ bits. For the second step, the k sites first use the algorithm in [11] to compute the degree of each vertex (this is essentially F_0) up to a factor of 2, using $O(kn \log n)$ bits of communication, and then they construct the set $H = \{v \in V \mid \text{degree}(v) \leq 2\sqrt{n}\}$. Next, $P_2, \ldots, P_k$ send P_1 all edges that are incident to a vertex in H. Note that each of P_i $(i = 2, \ldots, k)$

will send at most $O(n^{3/2})$ edges to P_1. Therefore, the total communication cost of the second step is bounded by $O(kn \log n + kn^{3/2} \log n) = O(kn^{3/2} \log n)$ bits. We conclude with the following theorem for diameter.

Theorem 5 *There exists a randomized protocol that approximates the diameter of a graph up to an additive error of* 2 *in the message-passing model. The protocol succeeds with probability* 0.99 *and uses* $O(kn^{3/2} \log^2 n)$ *bits of communication.*

7 Concluding Remarks

In this paper we show that exact computation of many basic statistical and graph problems in the message-passing model are necessarily communication-inefficient. An important message we want to deliver through these negative results, which is also the main motivation of this paper, is that a relaxation of the problem, such as an approximation, is necessary in the distributed setting if we want communication-efficient protocols. Besides approximation, the layout and the distribution of the input are also important factors for reducing communication.

An interesting future direction is to further investigate efficient communication protocols for approximately computing statistical and graph problems in the message-passing model, and to explore realistic distributions and layouts of the inputs.

One question which we have not discussed in this paper but is important for practice, is whether we can obtain round-efficient protocols that (almost) match the lower bounds which hold even for round-inefficient protocols? Most simple protocols presented in this paper only need a constant number of rounds, except the ones for bipartiteness and (approximate) diameter, where we need to grow BFS trees which are inherently sequential (require $\Omega(\Delta)$ rounds where Δ is the diameter of the graph). Using the sketching algorithm in [5], we can obtain a 1-round protocol for bipartiteness that uses $\tilde{O}(kn)$ bits of communication. We do not know whether a round-efficient protocol exists for the additive-2 approximate diameter problem that could (almost) match the $\tilde{O}(kn^{3/2})$ bits upper bound obtained by the round-inefficient protocol in Section 6.6.

References

[1] Hadoop Project. http://hadoop.apache.org/.

[2] http://research.microsoft.com/trinity.

[3] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 574–576, 1999.

[4] F. N. Afrati, A. D. Sarma, S. Salihoglu, and J. D. Ullman. Upper and lower bounds on the cost of a map-reduce computation. *CoRR*, abs/1206.4377, 2012.

[5] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 459–467. SIAM, 2012.

[6] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proc. ACM Symposium on Principles of Database Systems*, pages 5–14, 2012.

[7] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5):454–465, 2012.

[8] M.-F. Balcan, A. Blum, S. Fine, and Y. Mansour. Distributed learning, communication complexity and privacy. *Journal of Machine Learning Research - Proceedings Track*, 23:26.1–26.22, 2012.

[9] Z. Bar-Yossef. *The complexity of massive data set computations*. PhD thesis, University of California at Berkeley, 2002.

[10] P. Brown, P. J. Haas, J. Myllymaki, H. Pirahesh, B. Reinwald, and Y. Sismanis. Toward automated large-scale information integration and discovery. In *Data Management in a Connected World*, pages 161–180, 2005.

[11] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms*, 7(2):21, 2011.

[12] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1), 2008.

[13] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.

[14] P. Erds and A. Rnyi. On the evolution of random graphs. In *Publication of the mathematical institute of the hungarian academy of sciences*, pages 17–61, 1960.

[15] C. Estan, G. Varghese, and M. Fisk. Bitmap algorithms for counting active flows on high-speed links. *IEEE/ACM Trans. Netw.*, 14(5):925–937, Oct. 2006.

[16] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings*, (1), 2008.

[17] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[18] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[19] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Proc. International Symposium on Algorithms and Computation*, pages 374–383, 2011.

[20] Z. Huang, B. Radunović, M. Vojnović, and Q. Zhang. The communication complexity of approximate maximum matching in distributed data. *Manuscript*, 2013. http://research.microsoft.com/apps/pubs/default.aspx?id=188946.

[21] Z. Huang, K. Yi, and Q. Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *Proc. ACM Symposium on Principles of Database Systems*, pages 295–306, 2012.

[22] H. D. III, J. M. Phillips, A. Saha, and S. Venkatasubramanian. Efficient protocols for distributed classification and optimization. In *ALT*, pages 154–168, 2012.

[23] H. D. III, J. M. Phillips, A. Saha, and S. Venkatasubramanian. Protocols for learning classifiers on distributed data. *Journal of Machine Learning Research - Proceedings Track*, 22:282–290, 2012.

[24] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proc. ACM Symposium on Principles of Database Systems*, pages 41–52, 2010.

[25] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 938–948, 2010.

[26] P. Koutris and D. Suciu. Parallel evaluation of conjunctive queries. In *Proc. ACM Symposium on Principles of Database Systems*, pages 223–234, 2011.

[27] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[28] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA*, pages 85–94, 2011.

[29] G. Malewicz, M. H. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 135–146, 2010.

[30] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: a fast and scalable tool for data mining in massive graphs. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 81–90, 2002.

[31] J. M. Phillips, E. Verbin, and Q. Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2012.

[32] V. Rastogi, A. Machanavajjhala, L. Chitnis, and A. D. Sarma. Finding connected components on map-reduce in logarithmic rounds. *CoRR*, abs/1203.5387, 2012.

[33] A. A. Razborov. On the distributional complexity of disjointness. In *Proc. International Colloquium on Automata, Languages, and Programming*, 1990.

[34] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.

[35] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, pages 607–614, 2011.

[36] J. Tarui. Finding a duplicate and a missing item in a stream. In *TAMC*, pages 128–135, 2007.

[37] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[38] D. P. Woodruff and Q. Zhang. Tight bounds for distributed functional monitoring. In *Proc. ACM Symposium on Theory of Computing*, 2012.

[39] A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. IEEE Symposium on Foundations of Computer Science*, 1977.